Tarea 2

Problemas conceptuales

1. Ejercicio 6.8: The City of Zion (Kleinberg & Tardos, página 319).

(a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
Schedule-EMP(x_1, ..., x_n)
  Let j be the smallest number for which f(j) ≥ x_n
    (If no such j exists then set j = n)
  Activate the EMP in the n^th second
  If n − j ≥ 1 then
    Continue recursively on the input x_1, ..., x_{n−j}
      (i.e., invoke Schedule-EMP(x_1, ..., x_{n−j}))
```

Dado el anterior algoritmo con las siguientes especificaciones:

Entrada: Un arreglo [x_1, …, x_n] con n >= 1

Salida: Un valor w entero con w >= 0 devuelve el mayor numero de robot que el rayo EMP puede destruir en t tiempo

Se puede demostrar que no funciona correctamente el algoritmo con el siguiente ejemplo:

i	**1**	**2**	**3**	**4**	**5**
x_i	**0**	**3**	**0**	**15**	**4**
f(i)	**1**	**2**	**4**	**8**	**16**

Respuesta del algoritmo:

Lo primero que se realiza en encontrar el j más pequeño tal que f(j) >= X_n , el cual es j = 3 dado que el X_n = 4, posteriormente activa el EMP en el segundo 5, dando como resultado min(4, 16) = 4, luego como la resta n – j = 2, entonces sigue haciendo recursión pero ahora solo con un arreglo que cuenta hasta el segundo 2 y vuelve a hacer la misma comparación, como no hay f(j) >= X_n , entonces j = n, se activa el EMP que da como resultado min(2, 3) = 2, y como n – j = 0, deja de hacer recursión da como resultado **6** robots máximos destruidos.

Respuesta correcta:

En el caso anterior y dada la respuesta del algoritmo se nota claramente que la mejor activación del EMP para destruir la mayor cantidad de robots no es en el segundo 2 y 5, realmente debe ser en el segundo 4 pues min(15, 8) = 8 y posteriormente en el segundo 5 pues min(4, 1) = 1, dado como resultado máximo de robots destruidos **9**

(b) Give an efficient algorithm that takes the data on robot arrivals $x_1, x_2, \ldots, x_n$, and the recharging function $f(\cdot)$, and returns the maximum number of robots that can be destroyed by a sequence of EMP activations.

Para resolver el problema anterior donde se tenían las siguientes especificaciones:

Entrada: Un arreglo $[x_1, \ldots, x_n]$ con n >= 1

Salida: Un valor w entero con w >= 0 devuelve el mayor numero de robot que el rayo EMP puede destruir en t tiempo

Se propone la siguiente función objetivo: phi(n, j) donde n es el valor índice de los segundos pasados y j el valor que se tiliza en la función f(j)

Reformulacion:

Entrada: Un arreglo $[x_1, \ldots, x_n]$ con n >= 1

Salida: phi(N, J)

Planteamiento recursivo:

$$\phi(n, j) \begin{cases} 0 & n = N+1 \\ \uparrow\begin{pmatrix} \phi(n+1, j+1) \\ \phi(n+1, 0) + \min(x[n], f(j)) \end{pmatrix} & n \mathrel{!}= N+1 \end{cases}$$

Pseudocódigo:

algoritmo robots(n, j)

si (n , j) esta en memoria entonces:

respuesta = memoria(n,j)

si no:

si n = N + 1 entonces:

respuesta = 0

si no:

respuesta = máximo(robots(n + 1, j), robots(n + 1, 0) + min(X[n], f(j)))

respuesta = memoria (n , j)

retornar respuesta

2. Ejercicio 6.10: Large computing jobs (Kleinberg & Tardos, página 321).

(a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

In minute 1, choose the machine achieving the larger of a_1, b_1
Set $i = 2$
While $i \leq n$
  What was the choice in minute $i - 1$?
  If A:
    If $b_{i+1} > a_i + a_{i+1}$ then
      Choose *move* in minute i and B in minute $i + 1$
      Proceed to iteration $i + 2$
    Else
      Choose A in minute i
      Proceed to iteration $i + 1$
    Endif
  If B: behave as above with roles of A and B reversed
EndWhile

Dado el anterior algoritmo con las siguientes especificaciones:

Entrada: Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

Salida: Un valor x entero con x >= 1 devuelve la mayor cantidad de trabajo que se puede realizar entre las 2 maquinas

Se puede demostrar que no funciona correctamente el algoritmo con el siguiente ejemplo:

N	**1**	**2**	**3**	**4**
A	**10**	**3**	**10**	**20**
B	**5**	**1**	**20**	**5**

Respuesta del algoritmo:

Con el anterior ejemplo lo que el algoritmo primeramente hace es verificar cuál de las dos maquinas A y B hace el mayor trabajo en el minuto 1, en este caso es A llevando una cuenta de +10, posteriormente se para en la posición B[3] y se pregunta si la posición A[2] + A[3] es menor, como si lo es entonces hace un cambio a la maquina B, ahora posicionados en la posición B[3] entonces se lleva una cuenta de +20, y pregunta si A[4] es mayor que B[3] + B[4] como no entonces seguimos con la maquina B llevando al final una cuenta de +10, Al finalizar la ejecución del algoritmo el resultado de las sumas da 35.

Respuesta correcta:

En el anterior caso y dada la respuesta del algoritmo se puede notar que claramente no es la respuesta correcta dado que si se quiere obtener la mayor cantidad de trabajo que se realiza en las 2 maquinas realmente se debe usar únicamente la maquina A en todo momento que suma una cantidad de 43 y no 35 como el resultado anterior

(b) Give an efficient algorithm that takes values for $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ and returns the *value* of an optimal plan.

Para resolver el problema anterior donde se tenían las siguientes especificaciones:

Entrada: Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

Salida: Un valor x entero con x >= 1 devuelve la mayor cantidad de trabajo que se puede realizar entre las 2 maquinas

Se propone la siguiente función objetivo: phi (n, t) donde n es el índice de los arreglos con n >= 1 y t es si estoy en el la maquina A o B, con 1 y 0 respectivamente, devolviendo como resultado el máximo trabajo que se puede realizar.

Reformulacion:

Entrada: Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

Salida: phi (N, T)

Planteamiento recursivo:

$$\phi(n,t)\begin{cases} 0 & \text{if } n == N+1 \text{ o } n = N+2 \\ & \text{else} \\ & \text{if } 1 \\ \uparrow\begin{pmatrix} \phi(n+1,t)+A[n] \\ \phi(n+2,0)+A[n] \end{pmatrix} & \\ & \text{else} \\ \uparrow\begin{pmatrix} \phi(n+1,t)+B[n] \\ \phi(n+2,1)+B[n] \end{pmatrix} & \end{cases}$$

Pseudocódigo:

algoritmo maquinas(n, t)

si (n , t) esta en memoria entonces:

respuesta = memoria(n,t)

si no:

si n = N + 1 o n = N + 2 entonces:

respuesta = 0

si no:

si t = 1 entonces:

respuesta = máximo (maquinas(n+1, t) + A[n], maquinas(n + 2, 0) + A[n])

si no:

respuesta = máximo (maquinas(n+1, t) + B[n], maquinas(n + 2, 1) + B[n])

respuesta = memoria (n , t)

retornar respuesta